Noname manuscript No.
(will be inserted by the editor)

Learning from Few Samples with Memory Network

Shufei Zhang · Kaizhu Huang · Rui Zhang · Amir Hussain ·

Accepted for publication in *Cognitive Computation. The final publication is available at Springer via https://doi.org/10.1007/s12559-017-9507-z*

Received: date / Accepted: date

Abstract Background. Neural Networks (NN) have achieved great successes in pattern recognition and machine learning. However, the success of a NN usually relies on the provision of a sufficiently large number of data samples as training data. When fed with a limited data set, a NN's performance may be degraded significantly.

Methods. In this paper, a novel NN structure is proposed called a Memory Network. It is inspired by the cognitive mechanism of human beings, which can learn effectively, even from limited data. Taking advantage of the memory from previous samples, the new model achieves a remarkable improvement in performance when trained using limited data. The memory network is demonstrated here using the Multi-Layer Perceptron (MLP) as a base model. However, it would be straightforward to extend the idea to other neural networks, e.g., Convolutional Neural Networks (CNN).

Results and Conclusions. In this paper the memory network structure is detailed, the training algorithm is presented and a series of experiments are conducted to validate the proposed framework. Experimental results show that the proposed model outperforms traditional MLP based models as well as other competitive algorithms in response to two real benchmark data sets.

Shufei Zhang
Xi'an Jiaotong-Liverpool University, SIP, Suzhou,215123, China
E-mail: zsftesila@gmail.com

Kaizhu Huang
Xi'an Jiaotong-Liverpool University, SIP, Suzhou,215123, China
E-mail: Kaizhu.Huang@xjtlu.edu.cn

Rui Zhang
Xi'an Jiaotong-Liverpool University, SIP, Suzhou,215123, China
E-mail: rui.zhang02@xjtlu.edu.cn

Amir Hussain
University of Stirling, Stirling FK9 4LA, UK
E-mail: ahu@cs.stir.ac.uk

Keywords Memory · Multi-layer Perceptron · Neural Network · Recognition · Prior Knowledge

1 Introduction

Neural Networks (NN), for example the Multi-Layer Perceptron (MLP) [1] [2], are widely used in pattern recognition, computer vision, and machine learning. Deep Neural Networks or deep learning models, are recently the most popular models used for many perceptron tasks including: listening (speech recognition), seeing (visual object recognition) and text understanding (natural language processing) [3] [4] [5] [6] [7] [8] [9]. To succeed, NNs and deep learning models usually require a sufficiently large training set in order to avoid overfitting [10]. When little training data is available, a NN's performance may be significantly limited. Moreover, to facilitate the training of a NN, input samples are usually assumed to be identically and independently distributed (i.i.d.) [11] [12] [13]. With the i.i.d. assumption, samples can be fed to a NN sequentially; this enables a stochastic gradient descent algorithm to be used for the training of the NN, which is both convenient and efficient [14] [15] [16]. In practice however, the assumption of an i.i.d data set may often be violated. The learning process followed by humans is not independent, rather it relies on previous knowledge. For example, if a child would like to learn how to run, the previous experience of walking provides some relevant knowledge, which can aid in learning to run more easily. Another more cognitive example would be where a native speaker of one language e.g. English tries to learn to speak another language e.g. French. Memory relating to the learning process of the first language would greatly aid in the learning of the second language. Both of these examples indicate that memory and previous knowledge are very important and could be used to improve learning practices.

Motivated by these examples, a novel neural network framework is proposed termed a Memory Network (MN). Employing a similar structure to a traditional NN (including input, hidden, and output layers), the MN introduces additional memory structures that can appropriately take advantages of previous knowledge learned from previous data sets in a new learning task. When only limited data are available, previous learned knowledge (stored in the memory network) could significantly benefit the training process for the present learning task. More specifically, memory of the network structures for the previous N training samples will be maintained. Depending on whether the present training sample shares the same category (true class label) or not, different constraints are enforced on the activations in each layer of the present network as were in the previous network. For example, if the previous sample shares the same true class label with the present sample, then similar activations of the same layers between the present network and previous network will be forced; otherwise, an attempt is made to enlarge the activations of the same layers between the present and previous network. One appealing feature

of the proposed MN is that, despite a seemingly complicated network, an efficient stochastic gradient descent algorithm can be readily applied to make the network easily optimized.

We list the main contributions as follows. 1) We build a novel model by adding the memory mechanism to the NN. 2) We study how the memory block can help learning: the memory block is able to encourage intra-class compactness and inter-class separability for latent features. 3) Our proposed model achieves a much better performance than other models on limited amount of data.

The rest of this paper is organized as follows. In Section 2, the notation used in the paper is presented together with an introduction of some necessary background knowledge. In Section 3, the Memory Network including its structure, model definition and detailed optimization are proposed. In Section 4, several experiments are formulated to test the performance of the proposed model and the results are discussed. Finally concluding remarks are made in Section 5.

1.1 Notation and Background

In this section, the basic principles of the conventional NN and the Back Propagation (BP) algorithm are reviewed and the notation to be used throughout the paper is also provided. Essentially, a NN is a stack of parametric non-linear and linear transformations [17]. Suppose a NN (with $L-1$ hidden layers) is trained to perform prediction in the scenario of classification. A NN will map a d-dimensional vector to a D-dimensional class space. The matrix X_0 denotes the input data matrix where each row of X_0 represents a sample vector ($X_{0,i}$ is the i^{th} sample vector with d dimensions). X_l indicates the activation of the l^{th} layer of the NN (where $l = 1, 2, ..., L-1$) and X_L denotes the output of the NN. Y is the vector containing the true class labels, each element of which is the true class label for a corresponding sample with D dimensions. The problem of the NN can be formulated as the following optimization problem:

$$\begin{aligned} \min_{W_{1:L}, b_{1:L}} \quad & \frac{1}{2}\|X_L - Y\|^2 \quad \text{s. t.} \\ X_l \ &= \ \sigma(X_{l-1}W_l + b_l), l = 1, ..., L-1 \\ X_L &= X_{L-1}W_L + b_L \end{aligned} \tag{1}$$

where $\sigma(.)$ is the element-wise sigmoid function for a matrix. For each element x of a matrix, the sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2}$$

In a NN, the sigmoid function is used to perform non-linear transformations; other functions can also be used for this such as $\max(0, x)$ or $\tanh(x)$.

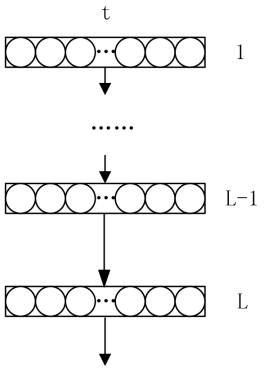


Fig. 1 The structure of conventional Neural Network

An illustrative example of a typical L-layer NN is shown in Figure 1, where X_l $(l = 1, 2, \ldots, L-1)$ represents the hidden layers. X_0 denotes the input to the NN, and X_L indicates the output of the NN. The aim is to learn the optimum parameters for $W_{1:L}$ and $b_{1:L}$. The common approach for this is to use BP with a stochastic gradient decent (SGD).

BP is an abbreviation for "backward propagation" of errors which is a common approach for training NNs with an optimization method such as gradient descent. The method calculates the gradient of a loss function with respect to all of the parameters of the network. The gradient is used in the optimization method which in turn uses it to update the parameters in order to minimize the loss function.

BP requires inputs with corresponding true class labels in order to calculate the loss function gradient. Therefore, it is considered as a supervised learning method, although it is also used in some unsupervised models such as auto-encoders. It is a generalization of the delta rule for multi-layered feed-forward networks, made possible by using the chain rule to iteratively calculate the gradients for each layer. Assuming that the activation function is differentiable, the gradients of parameters are shown below:

$$\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}X_L} &= 2(X_L - Y) \\
\frac{\mathrm{d}E}{\mathrm{d}X_l} &= (\frac{\mathrm{d}E}{\mathrm{d}X_{l+1}} \circ X_{l+1} \circ (1 - X_{l+1}))W_{l+1} \\
\frac{\mathrm{d}E}{\mathrm{d}W_l} &= X_{l-1}^T(\frac{\mathrm{d}E}{\mathrm{d}X_l} \circ X_l \circ (1 - X_l)) \\
\frac{\mathrm{d}E}{\mathrm{d}b_l} &= mean(\frac{\mathrm{d}E}{\mathrm{d}X_l} \circ X_l \circ (1 - X_l), 1)
\end{aligned} \tag{3}$$

where E is the value of the loss function and the gradients can be computed using the chain rule above. The element-wise product is represented by $\circ$ and $l = 1, 2, ...L$. In addition, $mean(., 1)$ denotes the average operation on the matrices.

2 Methods

This section introduces the proposed novel Memory Network in detail. First the structure of the MN is presented followed by discussion of the corresponding optimization algorithm.

2.1 Network Structure

The structure of the MN is plotted in Figure 2 and Figure 3. As can be seen, the structure of the MN consists of three parts: the present network, the memory block, and the square error operator, each of these will now be defined in detail.

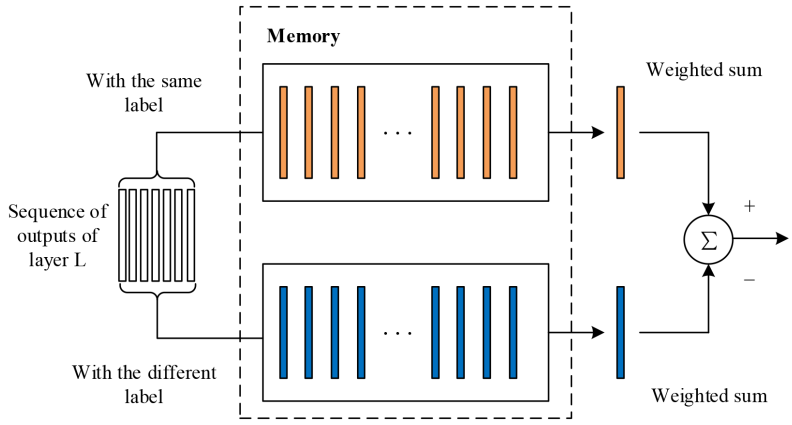


Fig. 2 Structure of memory

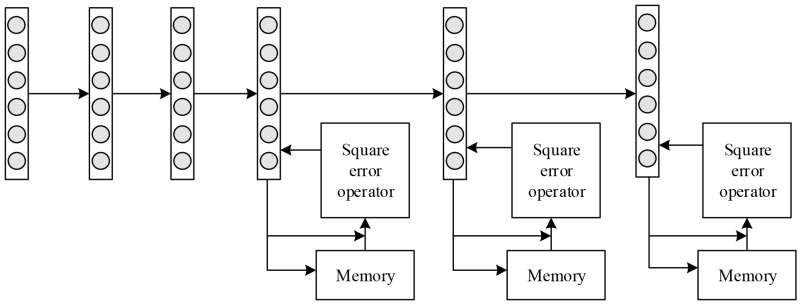


Fig. 3 Structure of memory network

Present Network The structure of the present network is the same as that of the traditional NN (consisting of an input layer, hidden layers and an output layer).

Memory Block The memory block in Figure 2 is divided into two sectors, one is used to store the outputs of layer L which share the same true class labels as the present output. The other is to store the outputs of layer L which have different true class labels to the present output. During the training procedure, the weighted sum is calculated, which is used to formulate the regularization term of each layer.

The purpose of the memory block is to exploit past knowledge (obtained from past samples) to help in the present learning process (of the present sample). There are two different cases: (1) if the present sample has the same true class label as the past sample, then the activations of the same layers in the present and past samples are made more similar; (2) if the present sample shares a different true class label from the past one, the activations of the top two layers for the present and past samples are made more different.

Square Error Operator The Square Error Operator is used to formulate the error between the present activation and previous one. Motivated by these processes, the training of the MN is described in the following subsections.

2.2 Model Formulation

In order to exploit past knowledge (obtained from previous examples) for present learning, the model of our proposed MN is designed as follows:

$$\begin{aligned} \min_{W_{1:L},b_{1:L}} \quad & \frac{1}{2}\|X_L^t - Y^t\|^2 + \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{N} k_j^i \|X_{L-j+1}^t - X_{L-j+1}^{t-i}\|^2 \quad \text{s. t.} \\ & X_l^t = \sigma(X_{l-1}^t W_l + b_l), \quad l = 1, ..., L-1, \\ & X_L^t = X_{L-1}^t W_L + b_L \end{aligned} \tag{4}$$

where X_l^t represents the activation of layer l for the present sample at time t, and Y^t is its corresponding true class label; X_l^{t-i} represents the activation of the previous i^{th} sample in layer l at time $t-i$, while Y^{t-i} describes the corresponding true class label for this specific sample. The matrix $\mathbf{k}$ (of size $p \times N$) is a coefficient matrix. Each element k_j^i is defined as a positive value, if $Y^t = Y^{t-i}$ (i.e., the previous i^{th} sample X_0^{t-i} shares the same true class label as the present sample X_0^t); otherwise it is a negative value. In other words, a positive k_j^i encourages more similarity between activations (in the $L-j+1$ layer) of the present learning (at t time) and the previous learning (at $t-i$ time); this is reasonable, since the present sample, X_0^t shares the same true class label as the previous sample X_0^{t-i}. Similarly, a negative k_j^i would enlarge

the difference between the activations of the current learning and the previous learning, since the present sample and the previous sample have a different true class label. It is also possible to adapt the value of k_j^i, depending on how deep the layer $L-j+1$ is. Usually, a deeper or upper layer (i.e., smaller j) is more important, suggesting that k_j^i should be set to a bigger value.

In short, on one hand, the optimization problem (4) attempts to minimize the loss at the current time t (when a sample X_0^t is input), i.e., the first term in (4); on the other hand, the proposed MN also tries to reduce (or enlarge) the difference in activations in the last p layers between the present network and previous networks, i.e., the memory loss in the second term of (4), depending on whether the present sample shares the same class label as the previous sample. Through this process, knowledge gained from previous training samples can be transferred to the present learning process, potentially leading to a network able to achieve a remarkable level of performance despite the limited number of training samples available.

2.3 Optimization

To solve the modified optimization problem above, it is still possible to rely on the BP algorithm, since the gradients with respect to the parameters can be easily computed from Eq. (4). For example, when p is set to 2, the gradients for output layer L could be calculated using:

$$\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}X_L} &= (X_L^t - Y^t) + \sum_{i=1}^{N} k_1^i (X_L^t - X_L^{t-i}) \\
\frac{\mathrm{d}E}{\mathrm{d}X_{L-1}} &= (\frac{\mathrm{d}E}{\mathrm{d}X_L} \circ X_L \circ (1 - X_L))W_L + \sum_{i=1}^{N} k_2^i (X_{L-1}^t - X_{L-1}^{t-i}) \\
\frac{\mathrm{d}E}{\mathrm{d}X_l} &= (\frac{\mathrm{d}E}{\mathrm{d}X_{l+1}} \circ X_{l+1} \circ (1 - X_{l+1}))W_{l+1} \\
\frac{\mathrm{d}E}{\mathrm{d}W_l} &= X_{l-1}^T (\frac{\mathrm{d}E}{\mathrm{d}X_l} \circ X_l \circ (1 - X_l)) \\
\frac{\mathrm{d}E}{\mathrm{d}b_l} &= mean(\frac{\mathrm{d}E}{\mathrm{d}X_l} \circ X_l \circ (1 - X_l), 1)
\end{aligned} \tag{5}$$

It is straightforward to extend the above cases to larger p's. With the above gradients, a BP can easily be conducted so that a local minimum can eventually be obtained for the memory network. The reason for setting p to 2 is to trade off space complexity and performance. The top 2 layers are chosen for the addition of memory since the top layers are more stable than the lower layers. In comparison to the traditional MLP, the proposed MN has memory blocks for the top the two layers, which can force the latent features to demonstrate intra-class compactness and inter-class separability (this is demonstrated through relevant experiments in the next section). For more general perspective, similar to [18], we assume that the latent features of different classes of penultimate

layer are sampled from the Gaussian distributions with different means and the same identical covariances. Our aim is to maximizes the arithmetic mean value of the Kullback-Leibler (KL) divergences between the different pairs of distributions and minimize the covariances while minimizing classification loss. Compared with p-laplacian regularized sparse coding method in [19], the regularization term in our proposed method is equivalent to p-laplacian regularization term when $p = 2$ and the weight function $w = 1$ for the same class, $w = -1$ for different classes.

3 Results

In this section, a series of experiments is conducted on two benchmark small-size data sets containing face and handwriting data.

3.1 Experimental Setup

The face data set contains 120 training samples [20] and the handwriting data set is a small portion of the MNIST data set [21]. The face data training and test sets are provided by [22]. Each model has been trained based on the same training data set; their respective performance in relation to the test set is then evaluated. Variable length handwriting data sets were produced by randomly selecting 50, 100, and 500 digits from the MNIST training set. Again, each model was trained based on the same training data set and their respective performance evaluated using the same test set. Five random training sets are formed for each set length. Each of the five is used to train each of the different models, which is then tested using the same test set. To compare the performance of the models the average classification accuracy over the five training sets is found. The performance of the proposed Memory Network is evaluated in comparison to; a conventional MLP, a Linear Support Vector Machine (linear-SVM) and its nonlinear version with the radial basis function (rbf) as the kernel (in short rbf-SVM).

The structure and parameters of the proposed models are set up differently for each data set. All of the models share the same depth with a total of 5 layers, i.e., 1 input layer, 3 hidden layers, and 1 output layer. The deep structure is exploited since deep networks are more flexible. For the face data set, the input-hidden-output units are respectively set to 100-300-100-40-15, and for the handwriting data set, the input-hidden-output units are 100-200-300-100-10. Both structures are tuned during the experiments. The value of p is set to 2, since the top layers are usually the most stable. The memory weights k_j^i are tuned from the set $\{0.0001, 0.001, 0.01, 0.1\}$. For the SVM, the trade-off parameter C and the width γ is tuned via cross validation.

3.2 Face Recognition with Different Pose

The face data set contains a total of 195 images of 15 persons [20]. Each person has 13 horizontal poses taken from -90 to 90 degrees at intervals of 15 degrees. The images were preprocessed by resizing them to 48×36 and redcuing the dimensions to 100 using Principal Component Analysis (PCA) [23]. Data set numbers $1-8$ are used as the training data sets and numbers $9-13$ are used as the test set.

Classifier	Accuracy (%)
Bilinear (Field) [24]	60.00
Style mixture (Singlet) [25]	70.00
Style mixture (Field) [25]	73.33
Nearest class mean [22]	60.00
FDA [22]	69.33
FBM [22]	74.67
linear-SVM	84.00
rbf-SVM	**85.33**
MLP	81.33
Memory Network	**85.33**

Table 1 Recognition rates of different models on face data. The proposed Memory Network and RBF-SVM significantly outperforms the other models. The other results (except Memory Network and conventional Neural Network) were copied from the associated papers due to the same setting.

Table 1 reports the performance (recognition rate) of different models. It can be noted that the test set shares very different poses from the training set which makes the problem very challenging. As observed, the novel Memory Network and rbf-SVM jointly achieve the best performances with 85.33%. More specifically, the proposed MN significantly improves on the performance of the MLP model by 4%. On the other hand, Fisher Discriminant Analysis (FDA) which is recognised as a state-of-the-art algorithm for face recognition, only achieved an error rate of 69.33% [22]. Moreover, other approaches such as the bilinear model, the style mixture model, the Field Bayesian Model and conventional NN all show significantly worse performance than the proposed Memory Network.

3.3 Handwriting Classification

Testing of the proposed MN model has also been performed using the digits section, MNIST, of the very famous handwriting data set, NIST [26], which has $60,000$ training samples and $10,000$ test samples. The samples chosen have all been size-normalized and centered in a fixed-size image (28×28). This experiment focuses on small sample sets from MNIST. Specifically, sets of 50, 100 and 500 samples are chosen randomly from the 60000 available samples. To increase training speed, the dimensions of the samples have been reduced

from 28×28 to 10×10. For testing, all test samples from the MNIST database are used ($10,000$ samples). The experiments were performed five times for each model and the average performance used to compare their relative accuracy.

#	Training Samples	50	100	500
	MLP	56.04 ± 1.60	67.76 ± 2.76	88.79 ± 0.87
	linear-SVM	63.10 ± 0.45	71.41 ± 2.96	86.69 ± 0.54
	rbf-SVM	64.56 ± 2.30	73.78 ± 1.78	89.20 ± 0.84
	Memory Network	$\mathbf{75.65 \pm 1.02}$	$\mathbf{81.60 \pm 1.92}$	$\mathbf{91.35 \pm 0.78}$

Table 2 Recognition rates (%) of different models on hand-writing data.

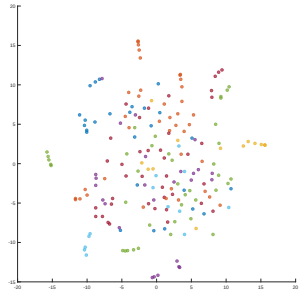
Comparing the performance of the proposed MN model with the conventional MLP, linear-SVM, and rbf-SVM models, Table 2 shows their performances (recognition rate). The proposed MN demonstrates a distinct relative improvement in performance as the number of training samples decreases. In particular, it can be noted that the proposed model achieves much better performance than the conventional MLP model on the 50-sample set (from 56.04% to 75.65%) and on the 100-sample set (from 67.76% to 81.60%). However, only marginal improvement is seen for the 500-sample set (from 88.79% to 91.35%). The proposed MN also outperforms both linear-SVM and rbf-SVM significantly. This experiment further validates the advantages of the proposed MN, especially when the training samples are limited.

3.4 Visualization

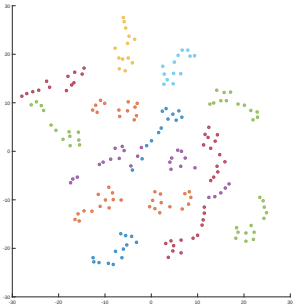
To obtain further insight into the proposed model MN, the latent features of the second top layer for the MNIST test set can be visualized. The traditional MLP is used as a baseline model. Figure 4 shows the feature visualizations of the trained MN and traditional MLP models based on the same face data. Figure 2 illustrates the latent features for the MN and traditional MLP models based on the same MNIST test set. It can clearly be seen that the memory blocks, i.e. the learned features of MN have better intra-class compactness and inter-class separability. This is the reason why the performance of the MN model is seen to be significantly better than that of the traditional MLP model.

3.5 Complexity Analysis

The complexity analysis is presented in this section. The results are described in Table 3, given that both the MLP and MN models are trained by the BP algorithm. With traditional MLP, the time complexity to calculate the gradients can be easily checked as $\mathcal{O}(nd + nD)$ during the training phase. For the proposed MN model, the time complexity is also $\mathcal{O}(nd + nD)$, the same as that for the traditional MLP. Here, n denotes the total number of

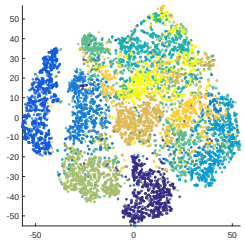


[Latent features of MLP for face data]

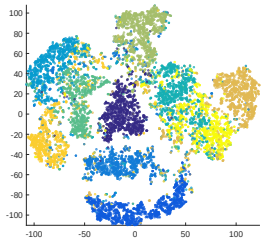


[Latent features of MN for face data]

Fig. 4 t-SNE embeddings of latent features for face dataset. Different colors indicate different classes of data. The graphs are better viewed in color.



[Latent features of MLP for MNIST]



[Latent features of MN for MNIST]

Fig. 5 t-SNE embeddings of latent features for MNIST dataset. Different colors indicate different classes of data. The graphs are better viewed in color.

training data, d and D represents the dimensions of the input and the output respectively. However, the space complexity of the proposed MN model is given as $\mathcal{O}(M + LN)$, larger than that of the traditional MLP, i.e., $\mathcal{O}(M)$. This is understandable, since the MN needs to memorize the previous outputs, where, M is total number of parameters stored temporarily for one batch, L and N represents the memory length and dimension of the latent output respectively. The number of parameters of the proposed MN and traditional MLP are the same for both tasks. Specifically, they are $64,955$ and $111,600$ respectively for both of the models in both the face and MNIST experiments. Overall, compared with MLP, the drawback of MN is its additional space complexity needed to store the previous latent outputs. Investigating this problem remains for future work.

Models	Time Complexity	Space Complexity
MLP	$\mathcal{O}(nd + nD)$	$\mathcal{O}(M)$
MN	$\mathcal{O}(nd + nD)$	$\mathcal{O}(M + LN)$

Table 3 Complexity analysis on MLP and MN

4 Conclusion

In this paper, a novel Memory Network which can appropriately take advantage of past knowledge is proposed. Specifically, a novel network with two parts: a memory block and a present part, both of which share the same structure have been constructed. The top p layers of the memory block and present part are connected and exploited to deliver the past knowledge to the present process. A modified stochastic optimization algorithm has been developed, which can efficiently optimize the proposed MN model. Experiments have been conducted using two small-size databases derived from face and handwriting data. Experimental results showed that the proposed model achieves the best performance in relation to both types of data set when compared to other competitive models. Our proposed model works in static situations and exploits the square error operator in memory block, in future, we will consider to modify it in dynamic situations with other kinds of error operators.

Declaration

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants performed by any of the authors.

Informed consent: Informed consent was obtained from all individual participants included in the study.

Acknowledgement

The paper was supported by National Science Foundation of China (NSFC 61473236), and Jiangsu University Natural Science Research Programme (14KJB520037).

References

1. D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.
2. S. Zhang and K. Huang. Learning from few samples with memory network. In *International Conference on Neural Information Processing*, pages 606–614. Springer, 2016.
3. F. Gao, Y. Zhang, J. Wang, J. Sun, E. Yang, and A. Hussain. Visual attention model based vehicle target detection in synthetic aperture radar images: a novel approach. *Cognitive Computation*, 7(4):434–444, 2015.
4. R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.
5. F. N. Iandola, S. Han, M. W Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
6. S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.
7. C. Lyu, K. Huang, and H.-N. Liang. A unified gradient regularization family for adversarial examples. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 301–309. IEEE, 2015.
8. M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.
9. Y. Sun, D. Liang, X. Wang, and X. Tang. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015.
10. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
11. E. Cambria and A. Hussain. *Sentic computing: a common-sense-based framework for concept-level sentiment analysis*, volume 1. Springer, 2015.
12. H. Chu, K. Huang, R. Zhang, and A. Hussian. Sdrnf: generating scalable and discriminative random nonlinear features from data. *Big Data Analytics*, 1(1):10, 2016.
13. Z. Malik, A. Hussain, and J. Wu. Extracting online information from dual and multi-data streams (in press). *Neural Computation and Applications, pages*, 2015.
14. K. Huang, H. Yang, I. King, and M. R. Lyu. *Machine learning: modeling data locally and globally*. Springer Science & Business Media, 2008.
15. K. Huang, H. Yang, I. King, and M. R. Lyu. Local learning vs. global learning: An introduction to maxi-min margin machine. In *Support vector machines: theory and applications*, pages 113–131. Springer, 2005.
16. H. Yang, K. Huang, I. King, and M. R. Lyu. Maximum margin semi-supervised learning with irrelevant data. *Neural Networks*, 70:90–102, 2015.
17. H. Wang and D.-Y. Yeung. Towards bayesian deep learning: A survey. *arXiv preprint arXiv:1604.01662*, 2016.

18. D. Tao, X. Li, X. Wu, and S. J. Maybank. Geometric mean for subspace selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):260–274, Feb 2009.
19. W. Liu, Z.-J. Zha, Y. Wang, K. Lu, and D. Tao. P-laplacian regularized sparse coding for human activity recognition. 63:1–1, 08 2016.
20. N. Gourier, D. Hall, and J. Crowley. Estimating face orientation from robust detection of salient facial features. In *International Conference on Pattern Recognition (ICPR)*, 2004.
21. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.
22. X.-Y. Zhang, K. Huang, and C.-L. Liu. Pattern field classification with style normalized transformation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1621–1626, 2011.
23. I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.
24. J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000.
25. P. Sarkar and G. Nagy. Style consistent classification of isogenous patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):88–98, Jan 2005.
26. P. J. Grother. Nist special database 19 handprinted forms and characters database. *National Institute of Standards and Technology*, 1995.